**Algoritmos y Programación Orientada a Objetos**

**Walter Bedoya Pérez**

**Jonathan Andrés Lopez Lopez**

**Tomas Ramírez Agudelo**

**ANÁLISIS DE PROBLEMA: JUEGO DE PELEA**

Se busca hacer un juego de pelea con el objetivo de derrotar la mayor cantidad de enemigos consecutivos, manejando una tabla de posiciones para guardar los jugadores con la mayor cantidad de victorias. El jugador iniciará registrando su nombre, y eligiendo su raza, ya sea luchador o mago, dependiendo de esta raza tendrá unas estadísticas particulares:

**Luchador:** Tendrá 2 ataques, el ataque básico con un multiplicador de daño, y el ataque potenciado sin multiplicador de daño, 500 de vida y 210 monedas.

**Mago:** Tendrá 2 ataques, el ataque básico con un multiplicador elevado de daño, y el ataque potenciado con multiplicador elevado de daño, 250 de vida y 75 monedas.

Para todas las razas iniciará con una cantidad de experiencia igual a 0, y un puntaje igual a 0, que irá aumentando conforme gane batallas.

Luego se desplegará un menú donde el jugador podrá elegir qué desea hacer, como:

- **Ir a la tienda:** donde podrá comprar pociones curativas con el dinero que tenga, hasta un máximo de 5 pociones (cada poción curará 25 de vida y costará 70 monedas).

-**Ver la mochila:** donde podrá ver la cantidad de pociones que tiene y si desea usarlas.

-**Ir al campo de entrenamiento:** donde podrá mejorar el potenciador de cada habilidad de a una unidad por el precio de 100 puntos de xp.

-**Ir a la batalla:** donde podrá empezar una pelea con un enemigo NPC aleatorio, aquí dentro se manejan los ataques por turnos, primero el jugador, luego el enemigo y así sucesivamente. Para el jugador se manejarán las decisiones dentro de la pelea por medio de un menú, que le dará la opción de elegir alguna acción:

- Escoger el tipo de ataque

- Abrir la mochila para curarse

- Huir (que contará como derrota)

- Escudarse, para cubrir una parte del daño del próximo ataque del enemigo.

El enemigo tendrá la opción de atacar, lo cual generará un ataque random en un rango de daño (8 a 20) multiplicado por un número que irá aumentando a medida que las victorias del jugador aumenten.

Cuando la vida del enemigo NPC llegue a 0, el jugador habrá ganado la batalla, entonces se le sumará un punto de victoria, obtendrá una cantidad de entre 70 y 130 monedas, una cantidad de 100 a 160 puntos de experiencia.

Pero si el jugador llega a 0 de vida, ya sea fuera o en medio de la batalla, la partida del jugador acabará, se pondrá en la tabla de posiciones su puntaje de victorias junto a su nombre, y se dará la opción de empezar una nueva partida desde cero.

**Requisitos Funcionales**

* Registrar jugador
* Comprar en la tienda
* Hacer turno del jugador
* Visualizar mi mochila
* Hacer turno del enemigo
* Finalizar batalla
* Finalizar partida
* Ver tabla de posiciones
* Entrenar jugador

| Nombre | R1 - Registrar jugador |
| --- | --- |
| Resumen | El sistema debe permitir registrar un jugador y elegir una raza para poder iniciar un juego |
| Entradas | Nombre del jugador, Raza elegida |
| Resultado | 1. El sistema muestra un mensaje dando la bienvenida al jugador  2. El sistema da las estadísticas de la raza elegida  3. El sistema le muestra el menú con la opción para ir a la tienda, entrar en combate, abrir la mochila o ir al campo de entrenamiento. |

| **Pasos** | **Métodos** | **Responsable** |
| --- | --- | --- |
| Crear jugador | def \_\_init\_\_(self, nombre: str): | Player |
|  |  |  |

| Nombre | R2 - Comprar en la tienda |
| --- | --- |
| Resumen | El sistema debe permitir que el jugador compre y agregue una poción a la mochila |
| Entradas | Cantidad de monedas, Cantidad de pociones |
| Resultado | 1. El sistema muestra un mensaje dando la bienvenida a la tienda del juego  2. El sistema muestra los items disponibles y su respectivo precio  3. El jugador elige el ítem que quiere comprar  4. En caso de que el jugador tenga 5 pócimas, el sistema muestra un mensaje que le aclara que no puede tener más de 5 pócimas  5. En caso de que el oro no le alcance al jugador para comprar el ítem, el sistema muestra un mensaje de que el oro no es suficiente |

| **Pasos** | **Métodos** | **Responsable** |
| --- | --- | --- |
| Definir Comprar Objeto | def comprar\_objeto(self, cantidad: int, jugador: Player) -> None: | Store |
| Comprar Objeto | store.comprar\_objeto(seleccion, jugador) | Store |

| Nombre | R3 - Hacer turno del jugador |
| --- | --- |
| Resumen | El sistema debe permitir que el jugador elija una de las opciones del menú de batalla |
| Entradas | Decisión (Usar habilidad), Abrir mochila (Para elegir poción y curarse), Utilizar su escudo, (Huir de la batalla) |
| Resultado | 1. El sistema le pide al jugador que seleccione una de las opciones  2. En caso de elegir atacar el sistema le muestra al jugador dos opciones de ataque, en caso de no elegir ninguna, el sistema elige el ataque básico  3. En caso de elegir escudarse, el jugador utiliza su escudo  4. En caso de elegir mochila, se abrira la mochila permitiendo elegir alguno de los ítems que tiene adentro (en caso de estar vacía, mostrará el mensaje de “Lo siento, su mochila está vacía”)  5. En caso de elegir huir, el jugador perderá automáticamente la partida |

| **Pasos** | **Métodos** | **Responsable** |
| --- | --- | --- |
| Seleccionar una opción | seleccion=int(input()) | Combate |
| Atacar | def atacar(self, eleccion): | Combate |
| Usar escudo | def usar\_escudo(self): | Combate |
| Usar poción | jugador.curarVida() | Combate |
| Huir | def huir(self): | Combate |

| Nombre | R4 - Visualizar mi mochila |
| --- | --- |
| Resumen | El sistema debe permitir que el jugador pueda revisar que hay dentro de su mochila. |
| Entradas | mochila, pociones, Pocion |
| Resultado | 1. El sistema le da la opcion de ver la mochila al jugador  2. En caso de elegir ver la mochila, le mostrará los ítems comprados al jugador  3. En caso de que esté vacía mostrará el mensaje: ”Lo siento, su mochila está vacía” |

| **Pasos** | **Métodos** | **Responsable** |
| --- | --- | --- |
| Abrir mochila  Elegir item | def \_\_init\_\_(self):  self.pociones: list[Pocion] = [] | BackPack |

| Nombre | R5 - Hacer turno del enemigo |
| --- | --- |
| Resumen | El sistema debe permitir que el enemigo realice una acción dentro de su turno |
| Entradas | atacar |
| Resultado | 1. El enemigo ataca |

| **Pasos** | **Métodos** | **Responsable** |
| --- | --- | --- |
| Atacar (Enemigo) | def atacar(self):  return random.randint(8, 25) \* (1 + self.nivel \* 0.5) | Enemigo |

| Nombre | R6 - Finalizar batalla |
| --- | --- |
| Resumen | El sistema debe permitir que se finalice la batalla y se determine quién ganó. |
| Entradas | jugador.vida, enemigo.vida |
| Resultado | 1. Si la vida del jugador llega a 0, entonces se da al jugador como perdedor y se ejecuta R7  2. Si el jugador decide huir, también se dará como perdedor y se ejecuta R7  3. En caso de que sea la vida del enemigo la que llega a 0, el jugador se determinará como ganador y podrá continuar al siguiente menú |

| **Pasos** | **Métodos** | **Responsable** |
| --- | --- | --- |
| Perder la vida | if jugador.vida <= 0: | Combate |
| Huir y perder | def huir(self):  return True | Combate |
| Eliminar enemigo | elif enemigo.vida <= 0: | Combate |

| Nombre | R7 - Finalizar partida |
| --- | --- |
| Resumen | El sistema debe permitir que se finalice la partida y se determine cuantas victorias acomuló el jugador |
| Entradas | jugador.vida, jugador.victorias |
| Resultado | 1. Si la vida del jugador llega a 0  1.1 El sistema determina al jugador como perdedor  2. En caso de que el jugador elija huir  2.2 El sistema determina al jugador como perdedor  3. El sistema suma las victorias al puntaje del jugador registrado |

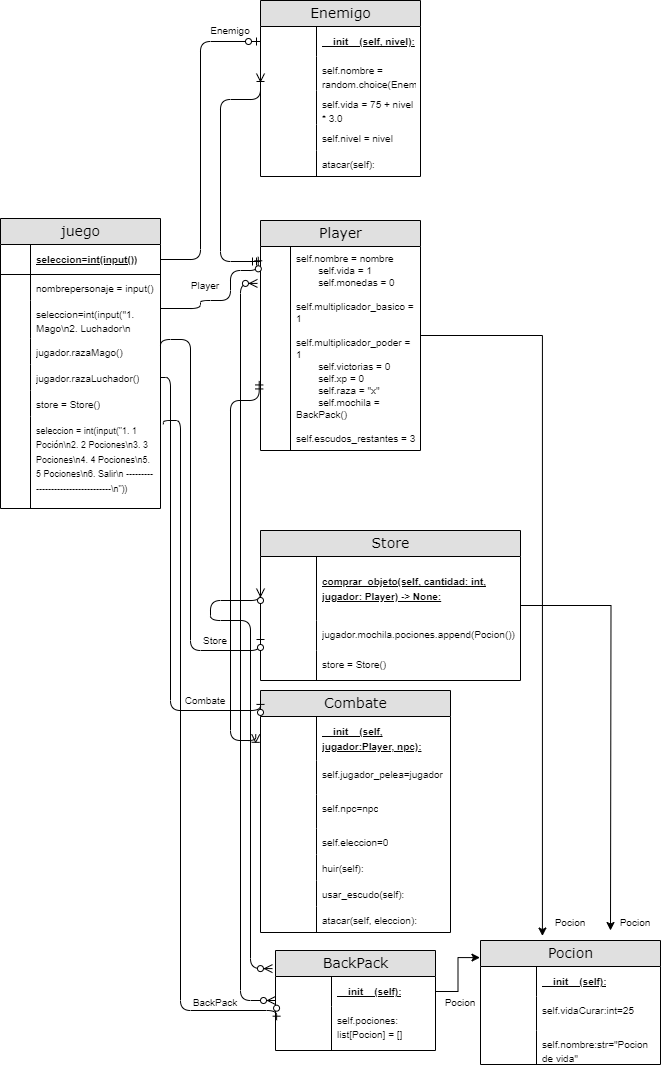
| **Pasos** | **Métodos** | **Responsable** |
| --- | --- | --- |
| Perder la vida y perder la partida | if jugador.vida <= 0:  partida = False | Combate |
| Huir y perder la partida | def huir(self):  return True  partida = False | Combate |
| Sumar puntaje | jugador.victorias += jugador.victorias | Tabla\_de\_puntuacion |

| Nombre | R8 - Ver tabla de posiciones |
| --- | --- |
| Resumen | El sistema debe permitir mostrar al jugador su puntaje obtenido y su posición respecto a los otros jugadores |
| Entradas | jugador.victorias |
| Resultado | 1. Muestra las victorias del jugador registrado y las de los otros jugadores |

| **Pasos** | **Métodos** | **Responsable** |
| --- | --- | --- |
| Mostrar puntaje | print(jugador.victorias) | Tabla\_de\_puntuacion |

| Nombre | R9 - Entrenar Jugador |
| --- | --- |
| Resumen | El sistema debe permitir al jugador entrenar para mejorar los multiplicadores de daño de alguno de sus ataques a cambio de 100 puntos de experiencia |
| Entradas | jugador\_pelea.multiplicador\_basico, jugador\_pelea.multiplicador\_poder, jugador.xp |
| Resultado | 1. Mejorar alguno de los ataques |

| **Pasos** | **Métodos** | **Responsable** |
| --- | --- | --- |
| Entrenar y mejorar algún ataque | jugador\_pelea.multiplicador\_basico+= 1  jugador\_pelea.multiplicador\_poder+= 1  jugador.xp -= 100 | Sala\_de\_entrenamiento |



<https://drive.google.com/file/d/1f1vR_fdrjYy0IJ9N0Nv_8S8UhPRF9ae6/view?usp=sharing>







https://www.canva.com/design/DAFvZxuY2Lo/zloAqD4UQeGxlu7\_rHgnvg/view?utm\_content=DAFvZxuY2Lo&utm\_campaign=designshare&utm\_medium=link&utm\_source=publishsharelink